WHAT IS CLAIMED IS:

1. A computer system for executing programs described in a machine language based on the stack architecture, comprising:

a data cache;

a data buffer that can hold data of variables;

a consolidated register file each entry of which is designed to hold data;

an advanced pointer stack each entry of which is designed to hold an entry address in said consolidated register file;

an instruction buffer having the construction of a FIFO queue each entry of which is designed to hold substance of an instruction;

an arithmetic/logic unit that is designed to execute arithmetic/logic operations; and

a load/store unit that can access said data cache and said data buffer;

wherein, in the case that an instruction including a pop operation from the operand stack is decoded, entry addresses in said consolidated register file, to the number of words to be popped, are popped from said advanced pointer stack;

in the case that an instruction including a push operation onto the operand stack is decoded, entries of said consolidated register file that have not been allocated are allocated, to the number of words to be pushed, and the addresses of said newly allocated entries of said consolidated register file are pushed onto said advanced pointer stack;

substance of each decoded instruction, together with the

popped / pushed entry addresses in said consolidated register file in the case that the instruction includes a pop / push operation, is written into said instruction buffer; and

unexecuted instructions held in said instruction buffer are to be executed on the principle of data drive.

2. The computer system according to claim 1, further comprising a completed pointer stack each entry of which is designed to hold an entry address in said consolidated register file;

wherein, when the instruction held in the head entry of said instruction buffer is/becomes ready to be completed, in accordance with the substance in said head entry of said instruction buffer, said completed pointer stack is manipulated so as to reproduce the operation that was applied on said advanced pointer stack in the course of decoding of said instruction, said head entry is dequeued; and

each entry of said consolidated register file whose address said completed pointer stack loses hold of on account of a pop operation is released from allocation.

3. The computer system according to claim 2, further comprising a free list that is designed to hold addresses of free entries of said consolidated register file;

wherein, in the initialized state, the addresses of all the entries of said consolidated register file are registered on said free list;

in the case that an entry of said consolidated register file needs to be allocated, an address of free entry of said consolidated register file is taken out of said free list; and

the address of each entry of said consolidated register file that is released from allocation is to be registered on said free list.

4. The computer system according to claim 2, further comprising an advanced pointer stack history file each entry of which is designed to hold contents of said advanced pointer stack;

each entry of said consolidated register file being designed to further hold a branch tag;

wherein, in decoding an instruction, a branch tag is written into each entry of said consolidated register file that is being allocated;

each time a conditional branch instruction is decoded, contents of said advanced pointer stack are written into an entry of said advanced pointer stack history file, and then, with an updated branch tag, speculative execution based on branch prediction is carried out; and

in the case that a branch prediction turns out to have missed, instructions decoded after the conditional branch instruction are invalidated, each entry of said consolidated register file in which a branch tag for instructions decoded after said conditional branch instruction is written is released from allocation, contents of said advanced pointer stack history file that were written when said conditional branch instruction was decoded are copied into said advanced pointer stack, and the process is resumed from the instruction at the right place.

5. The computer system according to claim 2,

said advanced pointer stack and said completed pointer stack being each constructed as a circular buffer;

wherein, in the case that the content of the bottom entry

holding an entry address in said consolidated register file is identical between said advanced pointer stack and said completed pointer stack, the data held in the entry of said consolidated register file indicated by said identical content can be spilt into said data buffer, with the hold of the entry address in said consolidated register file in said bottom entry removed both in said advanced pointer stack and in said completed pointer stack; and

said consolidated register file can be filled with data from said data buffer by allocating a free entry of said consolidated register file to said data, writing said data into said entry, and having the entry under the bottom entry holding an entry address in said consolidated register file hold the address of said entry of said consolidated register file into which said data is being written both in said advanced pointer stack and in said completed pointer stack.

6. The computer system according to claim 3,

said free list being constructed as a FIFO queue;

wherein, in accordance with a plurality of instructions decoded simultaneously, manipulation of said advanced pointer stack, allocation of entries of said consolidated register file, and writing of substances of said plurality of instructions into successive entries of said instruction buffer are to be conducted at a time; and

in accordance with substances held in a plurality of successive entries of said instruction buffer, manipulation of said completed pointer stack, and release of entries of said consolidated register file from allocation are to be conducted at a time.

7. A computer system for executing programs described in a machine language based on the stack architecture, comprising:

a consolidated register file each entry of which is designed to hold data;

an advanced pointer stack each entry of which is designed to hold an entry address in said consolidated register file;

an instruction buffer having the construction of a FIFO queue each entry of which is designed to hold substance of an instruction;

functional units each having an appropriate number of reservation stations; and

a common data bus through which data and their respective entry addresses in said consolidated register file are to be distributed among said consolidated register file and said functional units;

wherein, in the case that an instruction including a pop operation from the operand stack is decoded, entry addresses in said consolidated register file, to the number of words to be popped, are popped from said advanced pointer stack;

in the case that an instruction including a push operation onto the operand stack is decoded, entries of said consolidated register file that have not been allocated are allocated, to the number of words to be pushed, and the addresses of said newly allocated entries of said consolidated register file are pushed onto said advanced pointer stack;

substance of each decoded instruction, together with the popped / pushed entry addresses in said consolidated register file in the case that the instruction includes a pop / push operation, is written into said

instruction buffer;

substance of each instruction that is written into said instruction buffer is written into a free reservation station of a functional unit that is to execute the instruction, if necessary according to the type of the instruction;

the contents of each entry of said consolidated register file whose address is popped from said advanced pointer stack are read out, and if data is already written, the entry address and the data are to be put on said common data bus;

in each of said reservation stations holding substance of an instruction, each address of entry of said consolidated register file to hold source data is compared with entry addresses in said consolidated register file delivered through said common data bus, data is taken in if any matched, and said instruction is to be performed after required source data are fully arranged;

each of said functional units is to put, on said common data bus, each result data produced by executing an instruction that pushes an entry address in said consolidated register file onto said advanced pointer stack when decoded, together with the pushed entry address in said consolidated register file; and

in accordance with contents delivered through said common data bus, data are written in said consolidated register file.

8. The computer system according to claim 7, further comprising a completed pointer stack each entry of which is designed to hold an entry address in said consolidated register file;

wherein, when the instruction held in the head entry of the queue of said instruction buffer is/becomes ready to be completed, in accordance with the substance in said head entry of said queue, said completed pointer stack is manipulated so as to reproduce the operation that was applied on said advanced pointer stack in the course of decoding of said instruction, said head entry is dequeued; and

each entry of said consolidated register file whose address said completed pointer stack loses hold of on account of a pop operation is released from allocation.